COMPLAINT EXHIBIT 12
U.S. Patent No. 8,050,321 (H.265)

As demonstrated in the chart below, ASUS directly and indirectly infringes at least claim 8 of U.S. Patent No. 8,050,321 (the "'321 Patent"). ASUS directly infringes, contributes to the infringement of, and/or induces infringement of the '321 Patent by making, using, selling, offering for sale, and/or importing into the United States the Accused Products that are covered by one or more claims of the '321 Patent. The Accused Products are devices that decode H.265-compliant video. For example, ASUS Q543MV Notebook ("ASUS Q543MV") is a representative product for other ASUS devices that decode H.265-compliant video.

The ASUS Q543MV contains at least one video decoder that helps decode H.265-compliant video.[1] While evidence from the ASUS Q543MV is specifically charted herein, the evidence and contentions charted herein apply equally to the other ASUS Accused Products that decode H.265-compliant video.

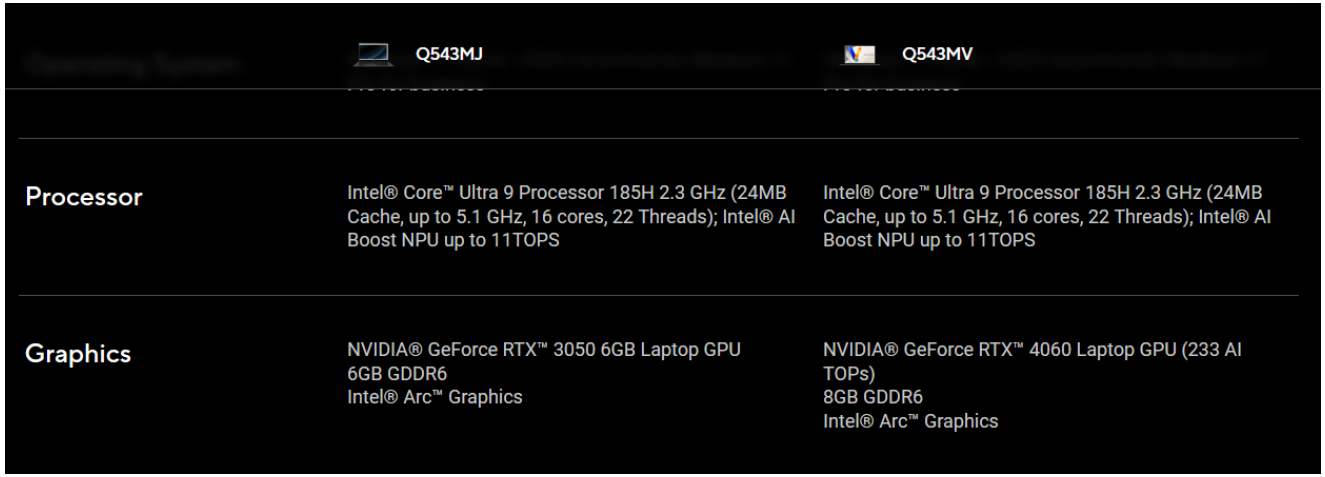
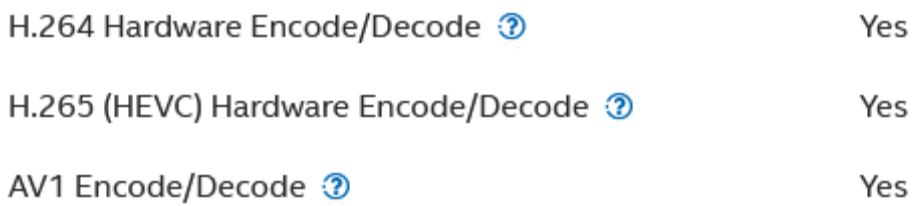
No part of this exemplary chart construes, or is intended to construe, the specification, file history, or claims of the '321 Patent. Moreover, this exemplary chart does not limit, and is not intended to limit, Nokia's infringement positions or contentions.

The following infringement chart includes exemplary citations to ITU-T Rec. H.265 (12/2016) High efficiency video coding (available at https://www.itu.int/rec/T-REC-H.265-201612-S/en) (the "H.265 Standard"). The cited functionality has been included in editions of the H.265 Standard since April 2013 and remains in current editions of the H.265 Standard. Any ASUS device that includes a decoder that practices the functionality in any of these editions of the H.265 Standard ("H.265 Decoder") practices the decoding claims of the '321 Patent. Thus, the ASUS Accused Products each practice the H.265 Standard and are covered by claims of the '321 Patent.

Nokia contends each of the following limitations is met literally, and, to the extent a limitation is not met literally, it is met under the doctrine of equivalents.[2]

[1] See, e.g., https://www.asus.com/us/laptops/for-home/everyday-use/asus-vivobook-pro-15-oled-q543/techspec/; https://www.intel.com/content/www/us/en/products/sku/236849/intel-core-ultra-9-processor-185h-24m-cache-up-to-5-10-ghz/specifications.html; https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new.

[2] This claim chart is based on the information currently available to Nokia and is intended to be exemplary in nature. Nokia reserves all rights to update and elaborate its infringement positions, including as Nokia obtains additional information during the course of discovery.

U.S. Patent No. 8,050,321	ASUS Accused Products
8. [A] A method for decoding a compressed video sequence,	Each of the Accused Products, such as the ASUS Q543MV, performs a method for decoding a compressed video sequence. For example, and without limitation, the Asus Q543MV uses hardware-accelerated decoding and includes an NVIDIA GeForce RTX 4060 Laptop graphics processing unit ("GPU") and an Intel Core Ultra 9 Processor 185H.  Source: https://www.asus.com/us/laptops/for-home/everyday-use/asus-vivobook-pro-15-oled-q543/techspec/ (last accessed March 6, 2025).  Source: https://www.intel.com/content/www/us/en/products/sku/236849/intel-core-ultra-9-processor-185h-24m-cache-up-to-5-10-ghz/specifications.html (last accessed March 6, 2025)(specifications for Intel Core Ultra 9 185H).

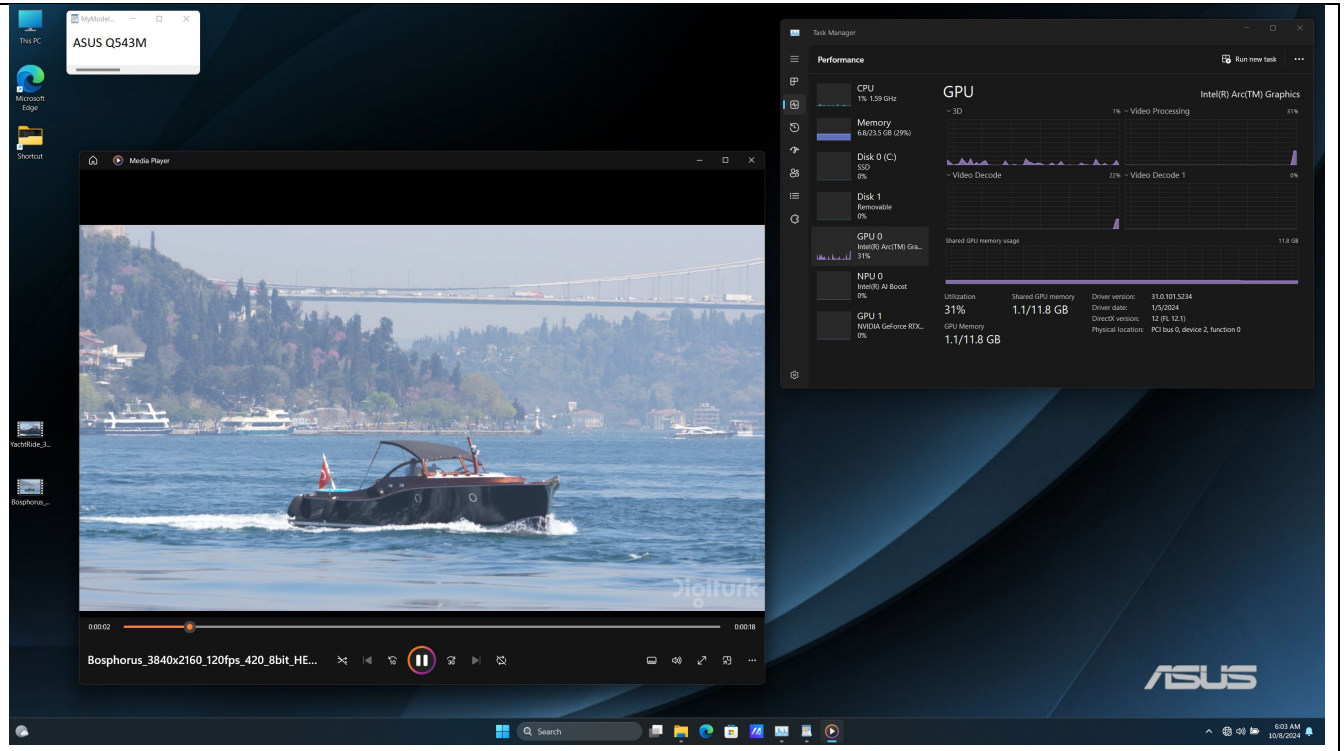
BOARD	FAMILY	NVENC Generation	Desktop/ Mobile	# OF CHIPS	Total # of NVENC	Max # of concurrent sessions	H.264 (AVCHD) YUV 4:2:0	H.264 (AVCHD) YUV 4:2:2	H.264 (AVCHD) YUV 4:4:4	H.264 (AVCHD) Lossless	H.265 (HEVC) 4K YUV 4:2:0	H.265 (HEVC) YUV 4:2:2	H.265 (HEVC) 4K YUV
GeForce RTX 4060 Laptop	Ada Lovelace	8th Gen	M	1	1	8	YES	NO	YES	YES	YES	NO	YES
GeForce RTX 4060	Ada Lovelace	8th Gen	D	1	1	8	YES	NO	YES	YES	YES	NO	YES

BOARD	FAMILY	NVDEC Generation	Desktop/ Mobile	# OF CHIPS	Total # of NVDEC	MPEG-1	MPEG-2	VC-1	VP8	VP9 4:2:0			H.264 (AVCHD) 4:2:0	
										8 Bit	10 Bit	12 Bit	8 Bit	10 Bit
GeForce RTX 4060 Laptop	Ada Lovelace	5th Gen	M	1	1	YES	YES	YES	YES	YES	YES	YES	YES	NO

H.265 (HEVC) 4:2:0			H.265 (HEVC) 4:2:2		H.265 (HEVC) 4:4:4			AV1	
8 Bit	10 Bit	12 Bit	8 Bit	10 Bit	8 Bit	10 Bit	12 Bit	8 Bit	10 Bit
YES	YES	YES	NO	NO	YES	YES	YES	YES	YES

Source: https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new (last accessed March 6, 2025)(row for 4060 Laptop GPU).

For example, an ASUS Q543MV was used to play back an H.265-compliant video.



Source: Screenshot of ASUS Q543MV playing back an H.265-compliant video.

For example and without limitation, the H.265 Standard specifies the following regarding the decoding process. The following specifications provide further evidence of how each of the Accused Products operates:

3 Definitions

. . .

3.30 coded video sequence (CVS): A sequence of *access units* that consists, in *decoding order*, of an *IRAP access unit* with NoRaslOutputFlag equal to 1, followed by zero or more *access units* that are not *IRAP access units* with NoRaslOutputFlag equal to 1, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *IRAP access unit* with NoRaslOutputFlag equal to 1.

	NOTE – An IRAP access unit may be an IDR access unit, a BLA access unit, or a CRA access unit. The value of NoRaslOutputFlag is equal to 1 for each IDR access unit, each BLA access unit, and each CRA access unit that is the first access unit in the bitstream in decoding order, is the first access unit that follows an end of sequence NAL unit in decoding order, or has HandleCraAsBlaFlag equal to 1. . . . **3.39 decoded picture**: A *decoded picture* is derived by decoding a *coded picture*. . . . **3.41 decoder:** An embodiment of a decoding process. . . . **3.43 decoding order**: The order in which *syntax elements* are processed by the *decoding process*. **3.44 decoding process**: The process specified in this Specification that reads a *bitstream* and derives *decoded pictures* from it. (ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 4 – 7).
[B] the method comprising: decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence;	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence. For example and without limitation, the H.265 Standard specifies decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence. For example, according to the H.265 Standard, each of the Accused Products receives a bitstream comprising one or more coded video sequences (CVSs) which are made up of access units, where each access unit comprises exactly one coded picture. For example, each of the Accused Products may receive CVSs, each starting with an independently decodable intra random access point (IRAP) picture which may be decoded without prediction from any other picture of the sequence. For example, one

example of an IRAP picture is an instantaneous decoding refresh (IDR) picture which has network abstraction layer (NAL) unit type equal to IDR_W_RADL or IDR_N_LP. For example, each of the Accused Products may receive an IDR picture, which is the first picture of a CVS in decoding order. Each of the Accused Products may receive a CVS in which an IRAP picture is followed in decoding order by zero or more non-IRAP pictures.

Each of the Accused Products may decode an indication of presence of an IDR picture, for example, by decoding a nal_unit_type value of 19 or 20 in the NAL Unit syntax.

The following specifications provide further evidence of how each of the Accused Products operates:

> **3 Definitions**
>
> For the purposes of this Recommendation | International Standard, the following definitions apply:
>
> **3.1 access unit**: A set of *NAL units* that are associated with each other according to a specified classification rule, are consecutive in *decoding order,* and contain exactly one *coded picture* with nuh_layer_id equal to 0.
>
> NOTE 1 – In addition to containing the video coding layer (VCL) NAL units of the coded picture with nuh_layer_id equal to 0, an access unit may also contain non-VCL NAL units. The decoding of an access unit with the decoding process specified in clause 8 always results in a decoded picture with nuh_layer_id equal to 0.
>
> NOTE 2 – An access unit is defined differently in Annex F and does not need to contain a coded picture with nuh_layer_id equal to 0.
>
> . . .
>
> **3.25 coded picture**: A *coded representation* of a *picture* containing all *coding tree units* of the *picture*.
>
> . . .
>
> **3.30 coded video sequence (CVS)**: A sequence of *access units* that consists, in *decoding order*, of an *IRAP access unit* with NoRaslOutputFlag equal to 1, followed by zero or more *access units* that are not *IRAP access units* with NoRaslOutputFlag equal to 1, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *IRAP access unit* with NoRaslOutputFlag equal to 1.
>
> NOTE – An IRAP access unit may be an IDR access unit, a BLA access unit, or a CRA access unit. The value of NoRaslOutputFlag is equal to 1 for each IDR access unit, each

	BLA access unit, and each CRA access unit that is the first access unit in the bitstream in decoding order, is the first access unit that follows an end of sequence NAL unit in decoding order, or has HandleCraAsBlaFlag equal to 1. . . . **3.39 decoded picture**: A *decoded picture* is derived by decoding a *coded picture*. . . . **3.43 decoding order**: The order in which *syntax elements* are processed by the *decoding process*. **3.44 decoding process**: The process specified in this Specification that reads a *bitstream* and derives *decoded pictures* from it. . . . **3.61 instantaneous decoding refresh (IDR) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is an *IDR picture*. **3.62 instantaneous decoding refresh (IDR) picture**: An *IRAP picture* for which each *VCL NAL unit* has nal_unit_type equal to IDR_W_RADL or IDR_N_LP. NOTE – An IDR picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each IDR picture is the first picture of a CVS in decoding order. When an IDR picture for which each VCL NAL unit has nal_unit_type equal to IDR_W_RADL, it may have associated RADL pictures. When an IDR picture for which each VCL NAL unit has nal_unit_type equal to IDR_N_LP, it does not have any associated leading pictures. An IDR picture does not have associated RASL pictures. **3.63 inter coding**: Coding of a *coding block*, *slice*, or *picture* that uses *inter prediction*. **3.64 inter prediction**: A *prediction* derived in a manner that is dependent on data elements (e.g., sample values or motion vectors) of one or more *reference pictures*. NOTE – A prediction from a reference picture that is the current picture itself is also inter prediction. . . . **3.67 intra random access point (IRAP) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is an *IRAP picture*.

	3.68 intra random access point (IRAP) picture: A coded *picture* for which each *VCL NAL unit* has nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive. NOTE – An IRAP picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be a BLA picture, a CRA picture or an IDR picture. The first picture in the bitstream in decoding order must be an IRAP picture. Provided the necessary parameter sets are available when they need to be activated, the IRAP picture and all subsequent non-RASL pictures in decoding order can be correctly decoded without performing the decoding process of any pictures that precede the IRAP picture in decoding order. There may be pictures in a bitstream that do not refer to any pictures other than itself for inter prediction in its decoding process that are not IRAP pictures. . . . **3.85 network abstraction layer (NAL) unit**: A *syntax structure* containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*. . . . **3.122 reference picture**: A *picture* that is a *short-term reference picture* or a *long-term reference picture*. NOTE – A reference picture contains samples that may be used for inter prediction in the decoding process of subsequent pictures in decoding order. . . . **3.172 video coding layer (VCL) NAL unit**: A collective term for *coded slice segment NAL units* and the subset of *NAL units* that have *reserved* values of nal_unit_type that are classified as VCL NAL units in this Specification. . . . (ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 4, 6 – 10, 13). **7.3.1.2 NAL unit header syntax**

	nal_unit_header( ) { — Descriptor **forbidden_zero_bit** — f(1) **nal_unit_type** — u(6) **nuh_layer_id** — u(6) **nuh_temporal_id_plus1** — u(3) }

(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at p. 32).

7.4.2.2 NAL unit header semantics

. . .

nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table 7-1.

. . .

Table 7-1 – NAL unit type codes and NAL unit type classes

nal_unit_type	Name of nal_unit_type	Content of NAL unit and RBSP syntax structure	NAL unit type class
. . .			
19 20	IDR_W_RADL IDR_N_LP	Coded slice segment of an IDR picture slice_segment_layer_rbsp( )	VCL
. . .			

. . .

NOTE 5 – An instantaneous decoding refresh (IDR) picture having nal_unit_type equal to IDR_N_LP does not have associated leading pictures present in the bitstream. An IDR picture having nal_unit_type equal to IDR_W_RADL does not have associated RASL pictures present in the bitstream, but may have associated RADL pictures in the bitstream.

(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 65 – 67).

Further, the evidence cited for claim limitation 8[A] applies to this claim limitation.

[C] starting the decoding of the video sequence from said first image frame of	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: starting the decoding of the video sequence from

the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame;	said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame. For example and without limitation, the H.265 Standard specifies starting the decoding of the video sequence from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame. For example, each of the Accused Products starts decoding of each CVS with an IRAP picture (for example an IDR picture) that does not refer to any pictures other than itself for inter prediction in its decoding process. The CVS is decoded without prediction from any image frame decoded prior to the first image frame. The following specifications provide further evidence of how each of the Accused Products operates: **3 Definitions** . . . **3.30 coded video sequence (CVS)**: A sequence of *access units* that consists, in *decoding order*, of an *IRAP access unit* with NoRaslOutputFlag equal to 1, followed by zero or more *access units* that are not *IRAP access units* with NoRaslOutputFlag equal to 1, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *IRAP access unit* with NoRaslOutputFlag equal to 1. NOTE – An IRAP access unit may be an IDR access unit, a BLA access unit, or a CRA access unit. The value of NoRaslOutputFlag is equal to 1 for each IDR access unit, each BLA access unit, and each CRA access unit that is the first access unit in the bitstream in decoding order, is the first access unit that follows an end of sequence NAL unit in decoding order, or has HandleCraAsBlaFlag equal to 1. . . . **3.61 instantaneous decoding refresh (IDR) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is an *IDR picture*. . . . **3.62 instantaneous decoding refresh (IDR) picture**: An *IRAP picture* for which each *VCL NAL unit* has nal_unit_type equal to IDR_W_RADL or IDR_N_LP. NOTE – An IDR picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or

may appear later in the bitstream. Each IDR picture is the first picture of a CVS in decoding order. When an IDR picture for which each VCL NAL unit has nal_unit_type equal to IDR_W_RADL, it may have associated RADL pictures. When an IDR picture for which each VCL NAL unit has nal_unit_type equal to IDR_N_LP, it does not have any associated leading pictures. An IDR picture does not have associated RASL pictures.

. . .

3.68 intra random access point (IRAP) picture: A coded *picture* for which each *VCL NAL unit* has nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.

NOTE – An IRAP picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be a BLA picture, a CRA picture or an IDR picture. The first picture in the bitstream in decoding order must be an IRAP picture. Provided the necessary parameter sets are available when they need to be activated, the IRAP picture and all subsequent non-RASL pictures in decoding order can be correctly decoded without performing the decoding process of any pictures that precede the IRAP picture in decoding order. There may be pictures in a bitstream that do not refer to any pictures other than itself for inter prediction in its decoding process that are not IRAP pictures.

. . .

3.116 random access skipped leading (RASL) picture: A *coded picture* for which each *VCL NAL unit* has nal_unit_type equal to RASL_R or RASL_N.

NOTE – All RASL pictures are leading pictures of an associated BLA or CRA picture. When the associated IRAP picture has NoRaslOutputFlag equal to 1, the RASL picture is not output and may not be correctly decodable, as the RASL picture may contain references to pictures that are not present in the bitstream. RASL pictures are not used as reference pictures for the decoding process of non-RASL pictures. When present, all RASL pictures precede, in decoding order, all trailing pictures of the same associated IRAP picture.

(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 6 – 8, 10).

Further, the evidence cited for claim limitations 8[A-B] applies to this claim limitation.

[D] decoding identifier values for image frames according to a numbering scheme; and	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: decoding identifier values for image frames according to a numbering scheme. For example and without limitation, the H.265 Standard specifies decoding identifier values for image frames according to a numbering scheme. For example, each of the Accused Products decodes from the bitstream a variable defined as picture order count (POC) that is associated with each picture and uniquely identifies the associated picture among all pictures in a CVS. For example, each of the Accused Products derives a POC value, which indicates to each of the Accused Products the position of the associated picture in output order relative to the output order positions of the other pictures in the same CVS that are to be output from the decoded picture buffer (DPB). The following specifications provide further evidence of how each of the Accused Products operates: **3 Definitions** For the purposes of this Recommendation | International Standard, the following definitions apply: . . . **3.25 coded picture**: A *coded representation* of a *picture* containing all *coding tree units* of the *picture*. . . . **3.30 coded video sequence (CVS)**: A sequence of *access units* that consists, in *decoding order*, of an *IRAP access unit* with NoRaslOutputFlag equal to 1, followed by zero or more *access units* that are not *IRAP access units* with NoRaslOutputFlag equal to 1, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *IRAP access unit* with NoRaslOutputFlag equal to 1. NOTE – An IRAP access unit may be an IDR access unit, a BLA access unit, or a CRA access unit. The value of NoRaslOutputFlag is equal to 1 for each IDR access unit, each BLA access unit, and each CRA access unit that is the first access unit in the bitstream in decoding order, is the first access unit that follows an end of sequence NAL unit in decoding order, or has HandleCraAsBlaFlag equal to 1. . . .

	3.40 decoded picture buffer (DPB): A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C. . . . **3.96 picture**: An array of *luma* samples in monochrome format or an array of *luma* samples and two corresponding arrays of *chroma* samples in 4:2:0, 4:2:2, and 4:4:4 colour format. NOTE – A picture may be either a frame or a field. However, in one CVS, either all pictures are frames or all pictures are fields. . . . **3.98 picture order count (POC)**: A variable that is associated with each *picture*, uniquely identifies the associated *picture* among all *pictures* in the *CVS*, and, when the associated *picture* is to be output from the *decoded picture buffer*, indicates the position of the associated *picture* in *output order* relative to the *output order* positions of the other *pictures* in the same *CVS* that are to be output from the *decoded picture buffer*. . . . **3.136 slice**: An integer number of *coding tree units* contained in one *independent slice segment* and all subsequent *dependent slice segments* (if any) that precede the next *independent slice segment* (if any) within the same *access unit*. **3.137 slice header**: The *slice segment header* of the *independent slice segment* that is a current *slice segment* or the most recent *independent slice segment* that precedes a current *dependent slice segment* in *decoding order*. **3.138 slice segment**: An integer number of *coding tree units* ordered consecutively in the *tile scan* and contained in a single *NAL unit*. **3.139 slice segment header**: A part of a coded *slice segment* containing the data elements pertaining to the first or all *coding tree units* represented in the *slice segment*. . . . (ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 6, 9, 11). **7.3.6 Slice segment header syntax** **7.3.6.1 General slice segment header syntax**

slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
if(!dependent_slice_segment_flag) {	
for(i = 0; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
slice_pic_order_cnt_lsb	u(v)

. . .

(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at p. 44).

7.4.7 Slice segment header semantics

7.4.7.1 General slice segment header semantics

When present, the value of the slice segment header syntax elements slice_pic_parameter_set_id, pic_output_flag, no_output_of_prior_pics_flag, slice_pic_order_cnt_lsb, short_term_ref_pic_set_sps_flag, short_term_ref_pic_set_idx, num_long_term_sps, num_long_term_pics and slice_temporal_mvp_enabled_flag shall be the same in all slice segment headers of a coded picture. When present, the value of the slice segment header syntax elements lt_idx_sps[i], poc_lsb_lt[i], used_by_curr_pic_lt_flag[i], delta_poc_msb_present_flag[i] and delta_poc_msb_cycle_lt[i] shall be the same in all slice segment headers of a coded picture for each possible value of i.

. . .

slice_pic_order_cnt_lsb specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the slice_pic_order_cnt_lsb syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits. The value of the slice_pic_order_cnt_lsb shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive. When slice_pic_order_cnt_lsb is not present, slice_pic_order_cnt_lsb is inferred to be equal to 0, except as specified in clause 8.3.3.1.

(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 94 – 95).

8.1.3 Decoding process for a coded picture with nuh_layer_id equal to 0

The decoding processes specified in this clause apply to each coded picture with nuh_layer_id equal to 0, referred to as the current picture and denoted by the variable CurrPic, in BitstreamToDecode.

. . .

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in clause 8.2.
2. The processes in clause 8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:
   – Variables and functions relating to picture order count are derived as specified in clause 8.3.1. This needs to be invoked only for the first slice segment of a picture.

. . .

(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 116 – 117).

8.3 Slice decoding process

8.3.1 Decoding process for picture order count

Output of this process is PicOrderCntVal, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction, and for decoder conformance checking (see clause C.5).

Each coded picture is associated with a picture order count variable, denoted as PicOrderCntVal.

When the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1, the variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows:

– Let prevTid0Pic be the previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL or SLNR picture.

– The variable prevPicOrderCntLsb is set equal to slice_pic_order_cnt_lsb of prevTid0Pic.

– The variable prevPicOrderCntMsb is set equal to PicOrderCntMsb of prevTid0Pic.

	The variable PicOrderCntMsb of the current picture is derived as follows:
	– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, PicOrderCntMsb is set equal to 0.
	– Otherwise, PicOrderCntMsb is derived as follows:
	if((slice_pic_order_cnt_lsb < prevPicOrderCntLsb) &&
	((prevPicOrderCntLsb − slice_pic_order_cnt_lsb) >=
	(MaxPicOrderCntLsb / 2)))
	PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb (8-1)
	else if((slice_pic_order_cnt_lsb > prevPicOrderCntLsb) &&
	((slice_pic_order_cnt_lsb − prevPicOrderCntLsb) > (MaxPicOrderCntLsb / 2)))
	PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb
	else
	PicOrderCntMsb = prevPicOrderCntMsb
	PicOrderCntVal is derived as follows:
	PicOrderCntVal = PicOrderCntMsb + slice_pic_order_cnt_lsb (8-2)
	NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since slice_pic_order_cnt_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0.
	The value of PicOrderCntVal shall be in the range of -2^{31} to $2^{31} - 1$, inclusive. In one CVS, the PicOrderCntVal values for any two coded pictures shall not be the same.
	. . .
	(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at p. 118).
	
	Further, the evidence cited for claim limitations 8[A-C] applies to this claim limitation.

[E] resetting the identifier value for the indicated first image frame of the independent sequence.	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: resetting the identifier value for the indicated first image frame of the independent sequence.

For example and without limitation, The H.265 Standard specifies resetting the identifier value for the indicated first image frame of the independent sequence.

For example, the H.265/HEVC Standard specifies that each of the Accused Products derives PicOrderCntVal that indicates the picture order count of the current picture. For example, for the IDR picture in a CVS, each of the Accused Products resets picture order count of the current picture (see for example, PicOrderCntVal) to 0 when an IDR picture is decoded at the beginning of an independent sequence.

The following specifications provide further evidence of how each of the Accused Products operates:

> **3 Definitions**
>
> For the purposes of this Recommendation | International Standard, the following definitions apply:
>
> . . .
>
> **3.30 coded video sequence (CVS)**: A sequence of *access units* that consists, in *decoding order*, of an *IRAP access unit* with NoRaslOutputFlag equal to 1, followed by zero or more *access units* that are not *IRAP access units* with NoRaslOutputFlag equal to 1, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *IRAP access unit* with NoRaslOutputFlag equal to 1.
>
> NOTE – An IRAP access unit may be an IDR access unit, a BLA access unit, or a CRA access unit. The value of NoRaslOutputFlag is equal to 1 for each IDR access unit, each BLA access unit, and each CRA access unit that is the first access unit in the bitstream in decoding order, is the first access unit that follows an end of sequence NAL unit in decoding order, or has HandleCraAsBlaFlag equal to 1.
>
> . . .
>
> **3.62 instantaneous decoding refresh (IDR) picture**: An *IRAP picture* for which each *VCL NAL unit* has nal_unit_type equal to IDR_W_RADL or IDR_N_LP.
>
> NOTE – An IDR picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each IDR picture is the first picture of a CVS in decoding order. When an IDR picture for which each VCL NAL unit has nal_unit_type equal to IDR_W_RADL, it may have associated RADL pictures. When an IDR picture for

which each VCL NAL unit has nal_unit_type equal to IDR_N_LP, it does not have any associated leading pictures. An IDR picture does not have associated RASL pictures.

. . .

(ITU-T Rec. H.265 (12/2016) High efficiency video coding, at pp. 4, 6 – 7).

8.3 Slice decoding process

8.3.1 Decoding process for picture order count

Output of this process is PicOrderCntVal, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction, and for decoder conformance checking (see clause C.5).

Each coded picture is associated with a picture order count variable, denoted as PicOrderCntVal.

When the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1, the variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows:

– Let prevTid0Pic be the previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL or SLNR picture.

– The variable prevPicOrderCntLsb is set equal to slice_pic_order_cnt_lsb of prevTid0Pic.

– The variable prevPicOrderCntMsb is set equal to PicOrderCntMsb of prevTid0Pic.

The variable PicOrderCntMsb of the current picture is derived as follows:

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, PicOrderCntMsb is set equal to 0.

– Otherwise, PicOrderCntMsb is derived as follows:

```
if( ( slice_pic_order_cnt_lsb < prevPicOrderCntLsb ) &&
    ( ( prevPicOrderCntLsb − slice_pic_order_cnt_lsb ) >=
    ( MaxPicOrderCntLsb / 2 ) ) )
  PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb  (8-1)
else if( (slice_pic_order_cnt_lsb > prevPicOrderCntLsb ) &&
    ( ( slice_pic_order_cnt_lsb − prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 )
    ) )
```

	PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb else PicOrderCntMsb = prevPicOrderCntMsb PicOrderCntVal is derived as follows: PicOrderCntVal = PicOrderCntMsb + slice_pic_order_cnt_lsb (8-2) NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since slice_pic_order_cnt_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0. The value of PicOrderCntVal shall be in the range of -2^{31} to $2^{31} - 1$, inclusive. In one CVS, the PicOrderCntVal values for any two coded pictures shall not be the same. . . . (ITU-T Rec. H.265 (12/2016) High efficiency video coding, at p. 118). Further, the evidence cited for claim limitations 8[A-D] applies to this claim limitation.